Metaheurística
Luís Felipe Ramos Ferreira

Como metaheurística foi desenvolvido o algoritmo de *Simulated Annealing*. Essa metaheurísticas é inspirada no processo físico de *annealing* utilizado na metalurgia pra alterar as propriedades físicas de algum metal. Essa abordagem pode ser utilizada em diversos problemas de natureza difícil, inclusive no TSP. O código da implementação pode ser encontrado nesse repositório, em particular a heurística descrita foi implementada no arquivo *simulated_annealing.py*.

A ideia principal da abordagem é testar caminhos novos aleatoriamente a cada iteração. Se o caminho novo for melhor que o atual, ele é mantido. Se for pior, ele é mantido com uma probabilidade proporcional à hiperparâmetros do algoritmo. O objetivo é fazer com que o algoritmo busque no espaço de soluções a solução ótima e sempre se movimente em direção à um mínimo. O uso de processos estocásticos auxilia para evitar que o algoritmo fique preso em um mínimo local e possa explorar novas áreas, podendo chegar assim mais perto do mínimo global.

Como hiperparâmetros inicias para essa bateria de testes apresentada neste documento, foram utilizados os valores de temperatura inicial igual a 10, $\delta_t = 0.9$ e limite inferior de temperatura igual a 0.10. Esse valores foram interesantes pois apresentaram um bom resultado em um tempo não muito alto para as instâncias testadas. No mundo real, diferentes valores devem ser testados para garantir que o objetivo principal ao ser utilizado o algoritmo seja atingido. Se o interesse principal for uma execução rápida, os parâmetros inciais podem ser bem diferentes de casos em que o desejo é o de um caminho do caixeiro de custo baixo, em troca de uma execução lenta.

Algorithm	Instance	Path Weight	Time (s)
sa	kroD100.tsp	32445	0.1660
sa	lin105.tsp	21930	0.1835
sa	att48.tsp	11916	0.0681
sa	kroB150.tsp	47631	0.2201
sa	berlin52.tsp	9419	0.0708
sa	kroA200.tsp	58072	0.2929
sa	rat195.tsp	4146	0.2827
sa	pr152.tsp	148754	0.2201
sa	st70.tsp	880	0.0967
sa	pr144.tsp	132181	0.2077
sa	pr124.tsp	123801	0.1748
sa	pr136.tsp	159539	0.1975
sa	kroB200.tsp	62789	0.2937
sa	kroB100.tsp	33466	0.1424
sa	kroA150.tsp	48895	0.2198
sa	kroA100.tsp	35865	0.1425
sa	pr107.tsp	75366	0.1497
sa	kroE100.tsp	34696	0.1406
sa	rat99.tsp	1786	0.1378
sa	kroC100.tsp	31168	0.1420
sa	pr76.tsp	142017	0.1082

Table 1: Path weights and times for various TSP instances using the sa (simulated annealing) algorithm